sources in a single application. In the case of queries that are parameterized with parameters being set by the portal, it in effect provides universal, client-side query joins, allowing the data from different databases to be combined together in a meaningful way. As a result, context portals allow users to attain a greater level of understanding of their data and thus to make better business decisions.

The dynamic object properties simplify development of data-driven applications, enabling domain experts with no programming experience other than spreadsheet usage to use the tool create powerful applications easily. Dynamic object properties have several advantages over static object properties which are hard-coded into the application when compiled. The bound to data. Furthermore, the bindings are applied to an entire class of object (all objects represented by the data element node) rather than simply to a single object (i.e., the calculated value for each object is dependent upon the row represented by the object). Another advantage is that it eliminates the specialized coding that would otherwise be required to perform such simple tasks as to make the color of an object dependent on the value of a column or set of columns from a data source. Further, more than one property of an object can be tied to a column in one row (or even a computation performed on all rows). In contrast to conventional development tools where each object is limited to a single link, the invention allows each column or row associated with an object to be linked to a separate property.

The generation of code from a scene graph eliminates the need for manual programming, thus allowing developers to concentrate on the problem domain rather than the tool itself. As a result, a domain expert with no programming experience can rapidly create powerful applications. In addition, the scene graph representation provides a powerful way of browsing an application's contents and making organizational changes to an application as easily as re-ordering the outline for a document in a word processor.

Additionally, the invention allows the representation of individual data points to be edited as generally as the top-level scene itself. This capability allows powerful applications to be created which support drill-down and exploratory navigation. This navigation allows users to attain a greater level of understanding of their data and thus to make better business decisions.

Other features and advantages will be apparent from the following description and the claims.

DESCRIPTIONS OF THE DRAWINGS

FIG. **1** is a multiple-document interface for editing one or more worlds.

FIG. **2** is a diagram illustrating an object inspector.

FIG. **3** is a diagram illustrating a property object model of an abstract base class VcPropertyBag.

FIG. **4** is a diagram illustrating two byte code execution classes, VCStmtCreateShape and VcStmtSetProperty.

FIG. **5** is a diagram illustrating a sample parsed property function.

FIG. **6** is a diagram illustrating an object model of a code generation process from a scene.

FIG. **7** is a flowchart illustrating a property sheet entry process.

FIG. **8** is a flowchart illustrating a process for visually manipulating an object.

FIG. **9** is a flowchart illustrating a process for byte code generation.

FIG. **10** is a flowchart illustrating a process for obtaining byte code execution statements for a node.

FIG. **11** is a flowchart illustrating a process for executing byte code.

FIG. **12** is a diagram illustrating a graph editing system.

FIG. **13** is a diagram illustrating a wormhole.

FIG. **14** is a diagram illustrating an object model of a wormhole.

FIGS. **15** and **16** are diagrams illustrating exemplary wormhole usages.

DESCRIPTION

A visual business intelligence system for building applications will now be described. In this system, a developer interactively builds a virtual world, whose building blocks include scenes, data sources, global parameters, and resources. A scene is a visual display of information much like a presentation slide, except that the information may be linked to data stored in a database. Within a scene, values resulting from a data source are represented graphically as user-defined data elements. Data sources are built with a block diagramming tool which generates one or more database queries. The queries can be SQL queries. Scenes are created with a drawing editor which transparently binds data sources to the graphical elements of the scenes. When the virtual world is completed, an execution image of the virtual world may be represented as byte code or a high level code which may be subsequently interpreted. The byte code representing the virtual world may be executed by a runtime control such as an ActiveX control. The runtime control may be embedded in a Web page or in any applications supporting the Active-X control such as Visual Basic, C or C++.

A scene may have one or more objects. Objects in a scene are described by a set of properties and may be assigned actions which are triggered by events such as clicking, user proximity to the scene, and mouse location. A viewpoint is a 3-dimensional location for viewing a scene. The location is described in terms of the user's current cursor X and Y offset from the center of the scene and the user's current zoom level (often referred to as magnification level). A viewpoint can be assigned a unique name and saved for facilitating navigation within a world. Named viewpoints can be used for manual navigation by the user or can be used as the target for an event-based jump action.

A data element is the graphical representation of a row resulting from a query. It is always associated with a layout and data source, and it is constructed much like a scene. It may contain any combination of objects, including layouts and wormholes to other scenes. Properties for objects making up a data element may not only reference scene parameters, but may also reference column names in the associated data source.

Scenes and data elements may be represented differently as a function of the user's zoom level. Each representation is called a level of detail (LOD), and is visually mutually exclusive from other LODs. For example, an object that starts off represented by a single dot may change into an icon as the user zooms closer, and then into a chart as the user zooms even closer.

Transition points between levels of detail are defined manually by the user in terms of zoom factors. A scene or data element may have N−1 transition points for N levels of detail. Visibility can also be controlled on a per-object basis by setting a "Visibility" property to a conditional expression such as

If(UserZoom>2, true, false)

Thus, transition regions may be created where alternate representations overlap for certain ranges of zoom levels